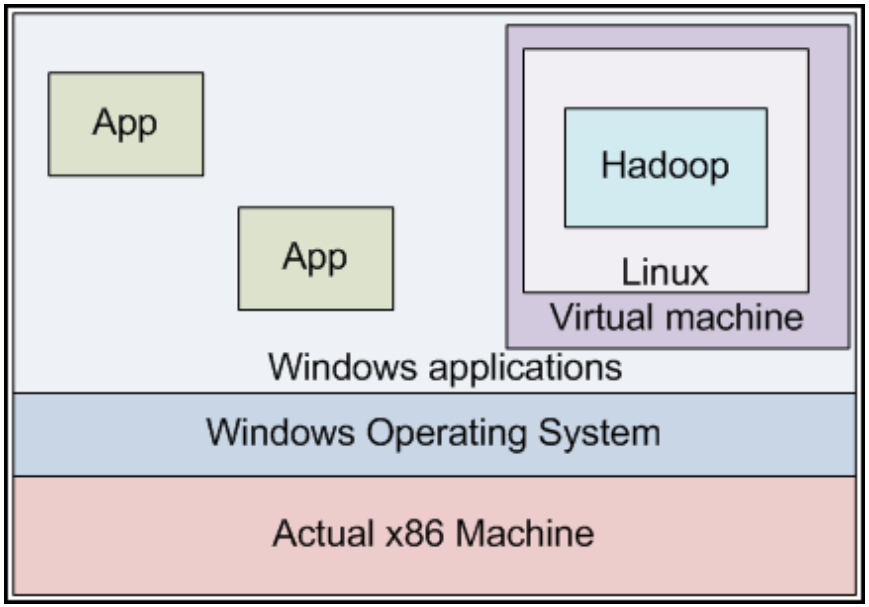
Hands on Programming

3:30-5:00pm

Steps for Starting the Hands-On

- Hadoop 20S released at the Hadoop Summit 2010
  - http://developer.yahoo.com/hadoop/
- Prepackaged VMWare image of Hadoop 20S
  - http://developer.yahoo.net/blogs/hadoop/2010/06/hadoop_020s_virtualmachine.html



Hadoop 20S Virtual Machine

- **Install VMware Player**
  - http://www.vmware.com/products/player/
- Setting up the Virtual Environment for Hadoop 0.20.S
  - Copy the Hadoop 0.20.S Virtual Machine into a location on your hard drive ~ 400MB
  - http://shared.zenfs.com/hadoop-vm-appliance-0-20-S.zip
- **Virtual Machine User Accounts**
  - Id: hadoop-user, Password: hadoop (sudo access)
  - Id: root, Password: root
- Hadoop Environment
  - Hadoop: 0.20.S (installed @ /usr/local/hadoop, /home/hadoop-user/ hadoop is symlink to install directory)
  - Pig: 0.7.0 (pig jar is installed @ /usr/local/pig, /home/hadoop-user/ pig-tutorial/pig.jar is symlink to one in install directory)

Using Hadoop in your Linux laptops

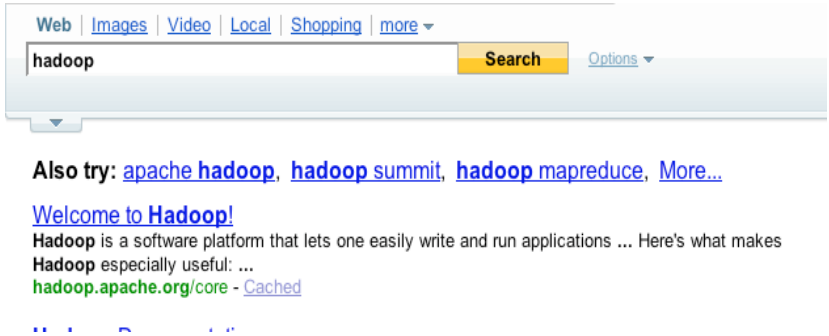
- cd ~/hadoop-0.20.2/bin
- Put $HADOOP_HOME as
  - ~/hadoop-0.20.2
- Put hadoop in your .bashrc/.cshrc..
  - export PATH=$PATH:$HADOOP_HOME/bin
- bash –x start-all.sh
  - Start all the namenode, jobtracker
- Access the Namenode using:
  - hadoop fs –ls /user/student
- Run Map Reduce program
  - bin/hadoop jar hadoop-examples-0.20.${yourversion}.jar pi 10 1000000
- Get Pig working
  - cd pig-0.7.0
  - java –cp pig-0.7.0-core.jar:$HADOOP_HOME/conf/ org.apache.pig.Main

Configuring VMWare image for Development

- Edit /etc/hosts
  - hadoop-user@hadoop-desk:sudo vi /etc/hosts
- Add the line
  - 91.189.88.30<TAB>us.archive.ubuntu.com
- Install javac using apt-get (Sun JDK 1.6)
  - hadoop-user@hadoop-desk:sudo apt-get install sun-java6-jdk

Exercise: NGrams

- Build a query-to-ngram mapping, that can be used as "Also try" in search.
- Simplify: Bigrams
- Inputs
  - A large document corpus



Bigrams

- Problem Description
  - Given a word, find out at most top N word pairs(2-gram) which consists of the word.
- The word pairs are output in decreasing order of their occurring times
- Input - a number of text files e.g.

  - w1 w2 w1 ...

- Ouput - a text file with lines as follows

  - w1 <w1 w2 5><w3 w1 3><w1 w4 2> ....
  - w2 <w1 w2 5><w2 w3 1>....

- Can you program this is Map Reduce Java and Hadoop Streaming
- Can you program this in Pig using Eval UDF

Bigrams: Step 1 - Map

- Generate all possible Bigrams
- Map Input: Text corpus
- Map computation
  - In each sentence, or each "A B"
  - Output (A, B), (B, A)
  - Partition & Sort by (A, B)

pairs.pl

```
while(<STDIN>) {
  chomp;
  $_ =~ s/[^a-zA-Z]+/ /g ;
  $_ =~ s/^\s+//g ;
  $_ =~ s/\s+$//g ;
  $_ =~ tr/A-Z/a-z/;
  my @words = split(/\s+/, $_);
  for (my $i = 0; $i < $#words - 1; ++$i) {
    print "$words[$i]:$words[$i+1]\n";
    print "$words[$i+1]:$words[$i]\n";
  }
}
```

Key-Value Separation in Map Output

```
$HADOOP_HOME/bin/hadoop  jar $HADOOP_HOME/hadoop-streaming.jar \
    -input myInputDirs \
    -output myOutputDir \
    -mapper org.apache.hadoop.mapred.lib.IdentityMapper \
    -reducer org.apache.hadoop.mapred.lib.IdentityReducer \
    -jobconf stream.map.output.field.separator=. \
    -jobconf stream.num.map.output.key.fields=4
```

Secondary Sort

```
$HADOOP_HOME/bin/hadoop  jar $HADOOP_HOME/hadoop-streaming.jar \
    -input myInputDirs \
    -output myOutputDir \
    -mapper org.apache.hadoop.mapred.lib.IdentityMapper \
    -reducer org.apache.hadoop.mapred.lib.IdentityReducer \
    -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \
    -jobconf stream.map.output.field.separator=. \
    -jobconf stream.num.map.output.key.fields=4 \
    -jobconf map.output.key.field.separator=. \
    -jobconf num.key.fields.for.partition=2 \
    -jobconf mapred.reduce.tasks=12
```

Bigrams: Step 1 - Reduce

- Reduce input
  - For each word A, a sorted list of phrases
  - All (A, B), then all (A, C) etc
- Reduce computation
  - Count all occurrences of (A, B)
- Reduce output
  - (A, [B, N])
  - Partitioned by A
  - Sorted by (A, B)

count.pl

```
$_ = <STDIN>; chomp;
my ($pw1, $pw2) = split(/:/, $_);
$count = 1;
while(<STDIN>) {
  chomp;
  my ($w1, $w2) = split(/:/, $_);
  if ($w1 eq $pw1 && $w2 eq $pw2) {
    $count++;
  } else {
    print "$pw1:", $count-(2**31-1), ":$pw2\n";
    $pw1 = $w1;
    $pw2 = $w2;
    $count = 1;
  }
}
print "$pw1:", $count-(2**31-1), ":$pw2\n";
```

Bigrams: Step 2

- Objective: Sort (A, [N, B]) by (A, N)
- Map: Identity function (/bin/cat)
- Partition by A
- Secondary Sort by N
- Reduce: Identity function
  - Also could be "Choose top N" (After sort, first N)

firstN.pl

```
$N = 5;
$FIX = 2**31 - 1;
$_ = <STDIN>; chomp;
my ($pw1, $count, $pw2) = split(/:/, $_);
$idx = 1;
$count += $FIX;
$out = "$pw1\t$pw2,$count;";
while(<STDIN>) {
  chomp;
  my ($w1, $c, $w2) = split(/:/, $_);
  $c += $FIX;
  if ($w1 eq $pw1) {
    if ($idx < $N) {
      $out .= "$w2,$c;";
      $idx++;
    }
  } else {
    print "$out\n";
    $pw1 = $w1;
    $idx = 1;
    $out = "$pw1\t$w2,$c;";
  }
}
print "$out\n";
```